Séparation d'haplotypes à partir de données de séquençage de troisième génération

Maxime Bridoux

Séparation d'haplotypes à partir de données de séquençage de 3e génération

Maxime Bridoux *

12 juillet 2018

Référence : stage effectué du 22 mai au 13 juillet au centre Inria Bretagne Atlantique sous la direction de Dominique Lavenier *.

Résumé

Les algorithmes d'assemblage actuels n'ont pas de performances satisfaisantes lorsque confrontés à des *reads* obtenus par des technologies de 3e génération issus de génomes diploïdes très hétérozygotes. On propose alors une méthode pour classer ces *reads* selon leur haplotype d'origine avant l'étape d'assemblage à partir d'un partitionnement sur les *k-mers* de ces *reads*.

Mots-clés : séquençage de 3e génération, génome diploïde hétérozygote, classification selon haplotype.

Classification ACM : I.5.3 (Clustering).

Table des matières

*. Univ Rennes, Inria, CNRS, IRISA, F-35000 Rennes, France

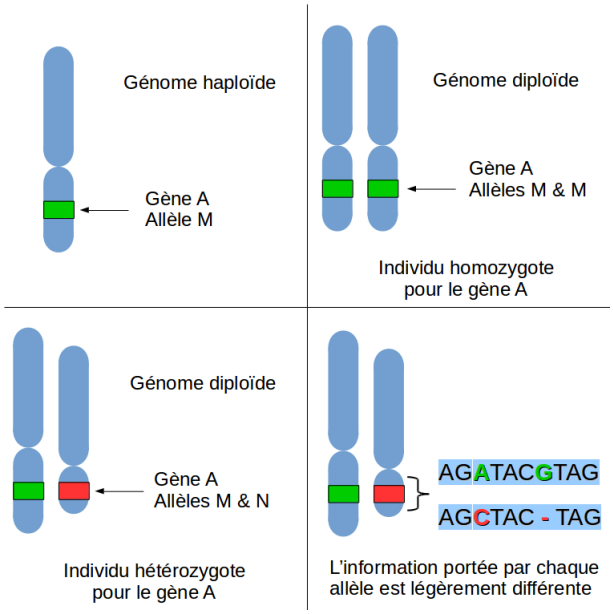


FIGURE 1 – Génome haploïde et génome diploïde.

1 Introduction

L'analyse du génome d'une espèce aide à comprendre le fonctionnement d'un organisme, d'une maladie ou d'un mécanisme biologique particulier. Ce génome dont on ne possède aucune information est obtenu en 2 étapes : tout d'abord, son séquençage extrait de nombreuses petites chaînes de nucléotides (chaque nucléotide est noté A, T, C ou G) appelées *reads*, dont par exemple ACGATCATGC. Il faut ensuite assembler ces *reads* ensemble pour obtenir le mot représentant le génome voulu.

Les *reads* produits dépendent grandement de la technologie de séquençage utilisée. Plus particulièrement, les technologies de 3e génération produisent des *reads* bien plus longs (20000 nucléotides au lieu de 200 en moyenne) que les technologies précédentes. Ces longs *reads* sont néanmoins obtenus avec des taux d'erreurs bien plus élevés (plus de 15% de différence avec le génome original au lieu de 0.1 à 1% pour les autres technologies). Les méthodes d'assemblage actuelles sont très sensibles aux erreurs ; d'autres méthodes ont été développées pour s'adapter au profil d'erreur des technologies de 3e génération.

On s'intéresse aux *reads* produits à partir de génomes diploïdes, c'est-à-dire un génome dont les chromosomes sont présents par paire (voir figure 1). Chaque chromosome possède donc l'information génétique sous deux formes très similaires : chaque version est appelée haplotype. Ces deux haplotypes possèdent les mêmes gènes, dont les différentes versions sont appelées allèles. Si les allèles sont les mêmes, on parle d'individu homozygote pour ce gène. Dans le cas contraire, on parle d'hétérozygotie. Plus généralement, le taux d'hétérozygotie désigne alors le rapport du nombre de paires de nucléotides qui diffèrent entre les deux haplotypes sur le nombre total de caractères.

L'assemblage sans génome de référence de tels génomes diploïdes très hétérozygotes (5 à 6% de différence entre les deux haplotypes) à partir de longs *reads* ne donnent pas de résultats satisfaisants. On cherche donc à séparer les deux haplotypes entre les étapes de séquençage et d'assemblage, i.e. classer les *reads* obtenus en deux catégories.

Pour ce faire, il sera d'abord question des particularités du séquençage de 3e génération, des techniques d'assemblage et des génomes diploïdes (section 2). Nous verrons ensuite comment mettre en relation certains *k-mers* (sous-mots de taille k) sélectionnés pour pouvoir classer les longs *reads* (section 3). Nous proposons deux méthodes qui s'appuient sur deux considérations différentes de l'information portée par les *k-mers*. Enfin, nous détaillerons le modalités d'expérimentation et l'efficacité de ces méthodes pour la résolution du problème. (section 4).

2 Description du domaine

2.1 Séquençage de 3e génération

Les techniques de séquençage de 3e génération font suite au séquençage nouvelle génération (NGS). Plutôt que cloner et casser des morceaux du génome pour en obtenir de petits fragments (*reads*), le génome est directement lu au niveau moléculaire, nucléotide par nucléotide [1]. Les *reads* obtenus par ces méthodes sont alors bien plus longs (20000 nucléotides en moyenne [2] au lieu de 200 nucléotides).

Les entreprises que sont PacBio et Oxford Nanopore Technologies ont développé leurs propres procédés de séquençage de 3e génération. La première technologie utilise le séquençage SMRT (Single Molecular Real Time sequencing) [1] qui construit la chaîne complémentaire à la chaîne voulue. L'ajout d'un nucléotide active une réaction fluorescente qui est détectée par un capteur de lumière [2]. La seconde technologie fait passer la chaîne à travers un nanopore dans lequel circule des ions [3, 4]. Le passage d'un nucléotide bloque les ions, ce qui permet d'identifier ce nucléotide grâce à la mesure du courant.

Le principal atout de ces nouvelles technologies de 3e génération est la taille des *reads* produits. En revanche celles-ci identifient un nucléotide à partir d'un signal (lumière ou courant) parfois très bruité et s'accompagnent donc de taux d'erreur bien plus élevés, de l'ordre de 15% et 38% respectivement pour les deux technologies [2, 4, 5, 6]. En comparaison, les technologies NGS ont des taux d'erreur de 0.1 à 1% [2].

Ces erreurs subissent néanmoins moins de biais que les technologies NGS. Ces dernières utilisent une amplification chimique qui peut conduire à sous-représenter certaines paires de nucléotides ou zones du génomes [7]. La petite taille de leurs *reads* tend également à éclipser les zones de redondance de nucléotides. Les *reads* de 3e génération rendent par leur plus grande taille la détection de ces zones plus facile. Leurs erreurs systématiques proviennent de limitations physiques (par exemple seuls six nucléotides à la suite peuvent rentrer dans un nanopore en même temps) ou de la mauvaise analyse du signal sur des zones où un même nucléotide est présent de nombreuses fois à la suite [4]. Les autres erreurs sont considérées comme uniformes.

2.2 Les techniques d'assemblage

Les *reads* obtenus vont servir à assembler un génome. Si l'on possède déjà un génome de référence de l'organisme voulu, on aligne les *reads* sur cette référence pour connaître la localisation de chaque nucléotide. On obtient pour chaque nucléotide de la référence quels nucléotides des *reads* s'alignent à cet endroit (leur nombre est appelé couverture). Malgré les taux d'erreurs importants, on obtient les particularités du génome séquencé via un consensus : le nucléotide

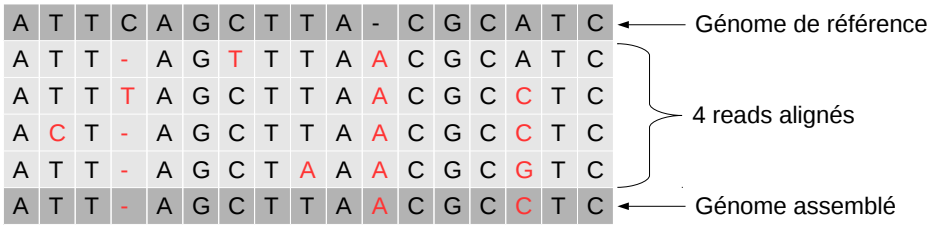


FIGURE 2 – Alignement et application du consensus avec génome de référence.

du génome séquencé à cet endroit est le nucléotide majoritaire parmi les longs *reads* [6].

La figure 2 représente l'alignement de quatre longs *reads* sur un génome de référence. Le génome assemblé est constitué pour chaque nucléotide du nucléotide majoritaire parmi les longs *reads*. La couverture est ici de 4.

L'assemblage final obtenu avec des longs *reads* est de bien meilleure qualité par rapport aux autres technologies, jusqu'à une erreur finale de moins d'une base sur un million [6]. Ce résultat est obtenu grâce à une couverture souvent très importante (100) possible grâce aux longs *reads* mais aussi par l'étape de consensus qui élimine facilement les erreurs de séquençage aléatoires [6].

L'assemblage *de novo* lui consiste en un assemblage sans génome de référence. Celui-ci est fait en assemblant les *reads* en petites portions du génome appelées *contigs*. Enfin, les *contigs* sont assemblés entre eux pour reconstituer le génome voulu. La longueur des *reads* des technologies de 3^{e} génération permet de réduire drastiquement le nombre de *contigs* : la structure du génome voulu est obtenue plus facilement. Le taux important d'erreur pose néanmoins problème pour les méthodes classiques telles que l'utilisation de graphe de de Bruijn [4]. On lui préfère les méthodes dites *overlap-layout-consensus* (OLC) [8].

2.3 Les génomes diploïdes

Contrairement à un génome haploïde caractérisé par une seule chaîne de nucléotides, un génome diploïde possède chacun de ses gènes en 2 exemplaires, appelés allèles. Ces deux chaînes différent d'un certain nombre de nucléotides ; ce taux de nucléotides mutés est appelé taux d'hétérozygotie. Ces nucléotides mutés posent problème lors de l'étape du consensus, où il n'y a pas de nucléotide majoritaire mais deux. Il faut également associer un nucléotide à un haplotype et sa mutation à l'autre haplotype.

Afin de pouvoir distinguer les erreurs de séquençage aux mutations, on identifie arbitrairement une chaîne comme base et l'autre comme sa mutation. Ceci permet de classer les mutations en 3 catégories. On va parler :

d'insertion lorsque la chaîne mutée possède un nucléotide mais pas l'autre chaîne ;

de délétion lorsque la chaîne mutée ne possède pas un nucléotide de l'autre chaîne ;

de substitution lorsque qu'un nucléotide de la chaîne de base a été remplacé par un autre nucléotide dans la chaîne mutée.

2.4 Problème rencontré

On possède donc un génome diploïde, dont le séquençage 3e génération donne de longs *reads*. S'il existe déjà un génome de référence, les *reads* peuvent être alignés sur ce génome. Le consensus permet alors d'identifier les sites de mutations en remarquant les endroits où il n'y pas un seul nucléotide majoritaire, mais deux en proportions proches [6]. Il reste donc dans ce cas à associer chaque mutation à un haplotype. On se concentre donc dans le cas particulier où il n'y a pas de génome de référence, celui de l'assemblage *de novo*.

Les algorithmes actuels d'alignement et d'assemblage sont paramétrés pour analyser les *reads* produits par une machine particulière et sont donc très sensibles par exemple à la répartition des erreurs de cette machine. Les méthodes *OLC* ont été utilisées pour faire face aux importants taux d'erreurs de séquençage des technologies de 3e génération. Celles-ci fonctionnent aussi pour des génomes diploïdes tels que le génome humain : celui-ci possède environ une mutation pour 3000 paires de nucléotides soit un taux d'hétérozygotie de 0.033 %. On s'intéresse ici à des génomes très hétérozygotes (5 à 6%), pour lesquels les méthodes *OLC* ne donnent pas de résultats satisfaisants.

L'objectif est donc de séparer les *reads* obtenus juste après le séquençage selon la chaîne (de base ou mutée) à laquelle ils appartiennent. Les deux chaînes pourront ainsi être assemblées séparément a posteriori.

3 Méthodes envisagées

3.1 Utilisation des k-mers

On souhaite séparer *reads* porteurs et non porteurs de mutations liés à l'hétérozygotie. Cela suppose de pouvoir identifier de tels sites de mutations. Sans alignement, il n'y a pas d'information précise de localisation d'un nucléotide sur la séquence globale. On va donc utiliser des *k-mers*, i.e les sous-mots de taille k des *reads*.

Pour chaque mutation, cela va donc constituer deux groupes de *k-mers*, chacun associé à une chaîne. Les erreurs de séquençage vont néanmoins également produire des *k-mers* parasites. Un critère intéressant pour séparer erreurs de séquençage des mutations entre les chaînes est la répartition des types d'erreurs. Par exemple, les mutations liées à l'hétérozygotie se repartissent chez les mammifères de la façon suivante :

— environ 90% de substitutions ;
— environ 10% d'insertions et de délétions.

Les techniques de séquençage de 3e génération possèdent elles des biais au niveau des répétitions de même nucléotide, qui sont d'insertion et de délétion en plus d'erreurs aléatoires. Ces deux répartitions sont donc très différentes.

Les *k-mers* porteurs d'erreurs de séquençage vont donc pouvoir être éliminés : parmi tous les *k-mers* produits à partir des *reads*, on va sélectionner au préalable les *k-mers* présents dans au moins un certain nombre de *reads* afin d'éviter ces erreurs de séquençage. Il reste une autre catégorie de *k-mers* : ceux qui ne contiennent pas de mutation. Il n'apportent pas d'information pour classer les *reads* qui les contiennent. Puisqu'il est difficile de les identifier, on les conserve dans un premier temps.

FIGURE 3 – Extrait du graphe produit par la première méthode pour des *k-mers* de taille $k = 10$ sur un génome de taille 1000, présentant de longs fils de *k-mers* reliés. Les k-mers en vert sont issus de la chaîne de base, les rouges de la chaîne mutée.

On va désormais tenter de classer les *k-mers* en deux catégories : pour chaque *read*, on prendre la classe majoritaire parmi les *k-mers* sélectionnés qu'il contient. Si le *read* ne contient pas de *k-mer* sélectionné, c'est qu'il ne porte pas de mutation (cas qui ne devrait pas arriver pour des *reads* de 20000 nucléotides pour 15% minimum de mutation). Pour classer ces *k-mers*, on va en conserver un que l'on attribue de façon arbitraire à une des catégories (qui malgré leur nom, sont interchangeables). On tente ensuite de grouper les *k-mers* ensemble pour savoir s'ils appartiennent ou non à la catégorie arbitrairement choisie.

3.2 Par rapport aux nucléotides portés

On essaie maintenant de mettre en relation les différents *k-mers* sélectionnés via les nucléotides portés. Pour ce faire, on utilise un graphe où les *k-mers* sont des nœuds. Deux *k-mers* sont mis en relation s'ils proviennent du même *read* ou s'ils ont une lettre de décalage. Cela forme un graphe, comme présenté en figure 3 avec plusieurs structures, notamment un long fil sur lequel se greffent des *k-mers* appareillés (voir figure 4), témoins d'une mutation ponctuelle. Le problème est donc de réussir à séparer *k-mers* de la base et de la mutation grâce à ce graphe.

On sait déjà quels *k-mers* sont ensemble à l'intérieur des *reads*, il faut ensuite s'aider des *reads* mis en relation grâce à la lettre de décalage. Le problème rencontré est qu'il n'y a pas de relation apparente entre deux *k-mers* avec deux lettres de décalage. En effet, deux *k-mers* de même catégorie peuvent être reliés tout comme deux *k-mers* de catégories différentes.

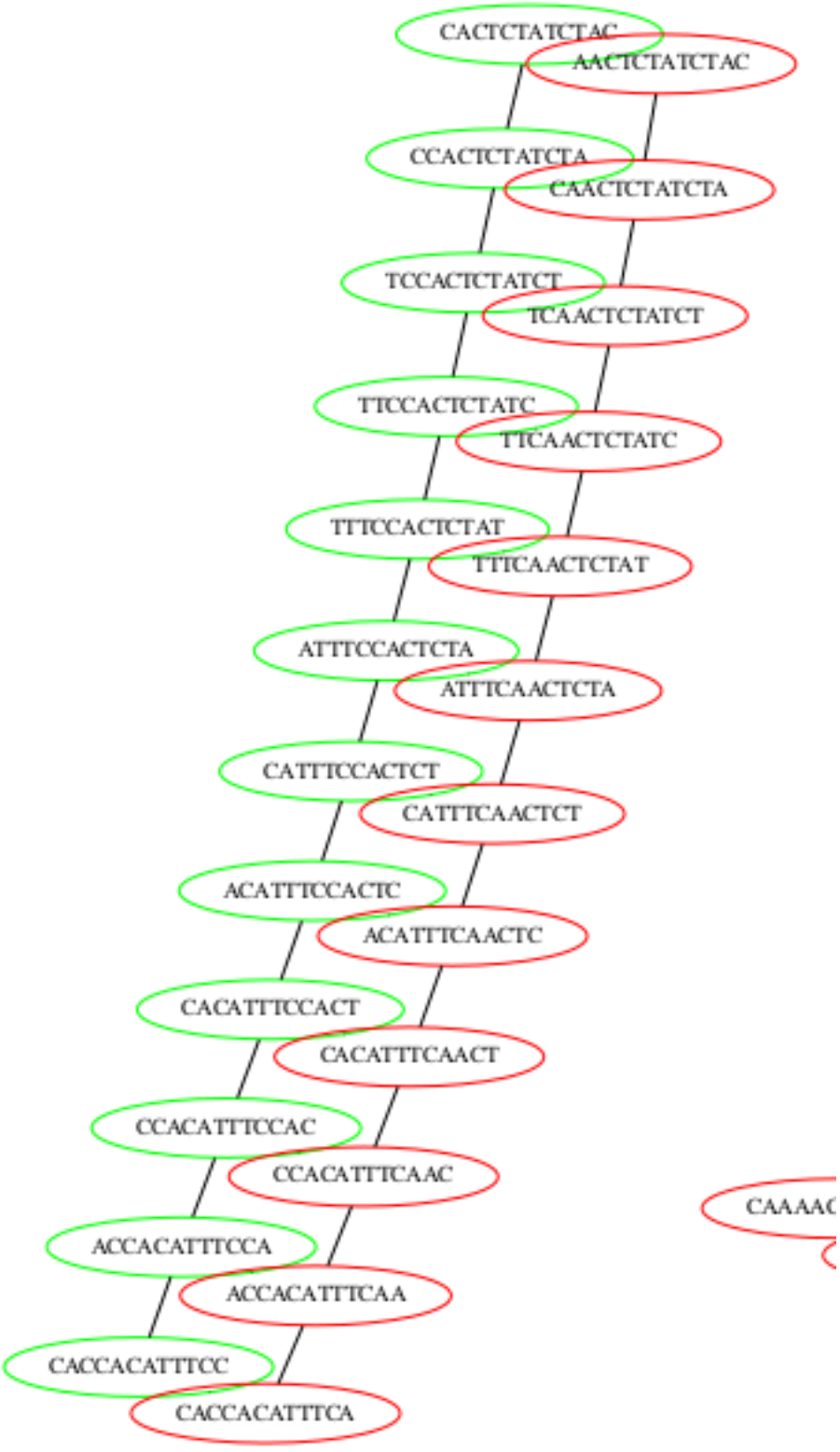
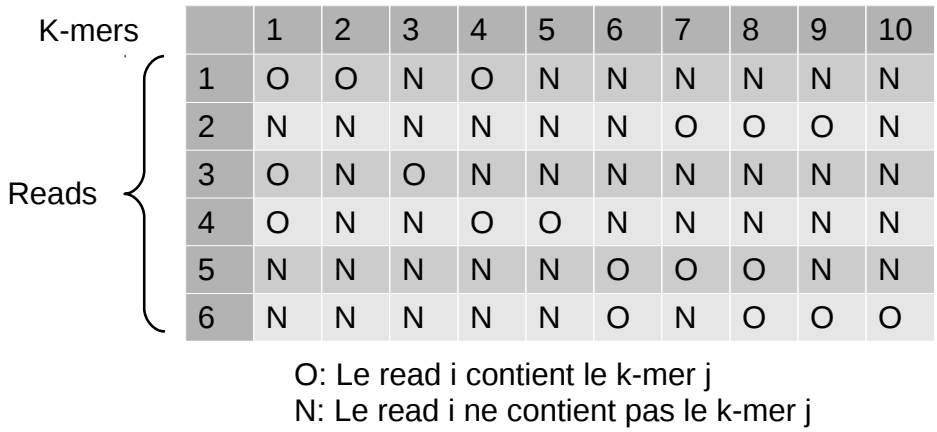


FIGURE 4 – Extrait du graphe produit par la première méthode pour des *k-mers* de taille $k = 10$ sur un génome de taille 1000, présentent un groupe de *k-mers* appareillés, témoins d'une substitution (un nucléotide C sur la chaîne de base à gauche est devenu un A sur les k-mers de la chaîne mutée à droite).



	1	2	3	4	5	6	7	8	9	10
1	O	O	N	O	N	N	N	N	N	N
2	N	N	N	N	N	N	O	O	O	N
3	O	N	O	N	N	N	N	N	N	N
4	O	N	N	O	O	N	N	N	N	N
5	N	N	N	N	N	O	O	O	N	N
6	N	N	N	N	N	O	N	O	O	O

FIGURE 5 – Exemple d'énumération de 10 *k-mers* contenus dans 6 *reads*.

3.3 Par rapport à la fréquence d'apparition dans les *reads*

Puisqu'on n'a pas réussi à exploiter l'information directement contenue par *k-mers*, on étudie ici une autre mise en relation, qui ne se base pas sur les nucléotides contenus. On a énuméré les *k-mers* dans chaque *read* pour pouvoir les sélectionner : on sait dire pour chaque *k-mer* s'il est contenu ou non dans un *read* particulier. On se sert de cette énumération pour savoir à quel point deux *k-mers* sont proches en comparant dans quels *reads* ils sont. Un exemple d'énumération est présenté en figure 5.

Une observation intéressante est que tous les *reads* d'une catégorie ne possèdent pas l'intégralité d'un groupe de *k-mers*. Réciproquement, d'autres *k-mers* ne sont présents dans aucun *read* de l'autre catégorie. Dans l'exemple de la figure 5, on peut notamment remarquer que les *reads* 1, 3 et 4 ne possèdent pas les *k-mers* 6 à 10 ; l'autre groupe de *reads* que sont les *reads* 2, 5 et 6 ne possèdent pas les *k-mers* 1 à 5.

Le problème est maintenant de séparer les *k-mers* en deux groupes, qui seront caractéristiques des deux groupes de *reads* recherchés. Plutôt que de regrouper les *k-mers* entre ceux qui sont présents dans les mêmes *reads*, on choisit de mettre en relation deux *k-mers* selon le nombre de *reads* dans lesquels ils sont tous les deux absents.

Soit le cas où les *reads* couvrent la même région et deux *k-mers* contenus par ces *reads*. On s'assure que ces *k-mers* ne sont pas contenus dans plus de la moitié des *reads*, car ils ne seraient alors pas porteurs de mutations et donc inutiles pour la classification. Les *k-mers* ont également une fréquence d'apparition minimale avec la sélection faite a priori :

s'il y a peu de *reads* qui ne contiennent pas ces *k-mers* (moins de 20% par exemple) : les deux *k-mers* appartiennent chacun à une catégorie. Dans l'exemple de la figure 5, cela peut correspondre au *k-mers* 1 et 6, dont seul le *read* 2 ne contient aucun des deux *k-mers* ;

s'il y a beaucoup de *reads* qui ne contiennent pas ces *k-mers* (50% et plus) : soit les deux *k-mers* appartiennent à la même catégorie (*k-mers* 1 et 4), soit ceux-ci n'ont pas une fréquence d'apparition suffisante (*k-mers* 5 et 10). Pour éviter ce cas, on prend pour les *k-mers* une fréquence d'apparition qui soit au moins la moitié du seuil de sélection de ce cas (50% ici), soit une fréquence d'apparition minimale d'au moins 25%, qui élimine les *k-mers* 5 et 10.

K-mers \ K-mers	1	4	6	7	8	9
1		3	1	1	0	1
4			2	2	1	2
6				3	3	3
7					3	3
8						3
9						

FIGURE 6 – Nombre de *reads* pour lesquels les deux *k-mers* sont simultanément absents pour chaque couple de *k-mers* ; les *k-mers* choisis sont ceux de la figure 5 avec au moins 25% d'apparition.

Les *k-mers* de l'exemple de la figure 5 ont donc été repris pour compter pour chaque couple de *k-mers* le nombre de *reads* dans lesquels les deux *k-mers* sont simultanément absents. La figure 6 regroupe ces résultats pour les *k-mers* ayant au moins 25% d'apparition parmi les *reads*. On remarque alors que si l'on récupère les couples ayant 50% ou plus de *reads* (soit 3 *reads* au minimum) dans lesquels ils ne sont pas présents, on est capable de constituer deux groupes de *k-mers* : les *k-mers* 1 et 4 d'un côté, et les *k-mers* 6, 7, 8 et 9 de l'autre.

Une fois les groupes de *k-mers* obtenus, on prend pour chaque *read* le groupe majoritaire parmi les *k-mers*.

4 Résultats obtenus

4.1 Préparation des données

Afin de pouvoir tester la démarche évoquée précédemment, il faut un génome constitué de deux chaînes, une base et une mutée. Il faut ensuite simuler le séquençage pour produire un groupe de *reads* par chaîne. Enfin, il faut mélanger ces *reads* qui serviront de base de travail, le but étant de pouvoir reconstruire les deux groupes précédents.

On crée ici une chaîne de lettres A, T, C, G de taille n où chaque lettre a 25% de probabilité d'être sélectionnée. Cette chaîne est ensuite copiée en mutant chaque lettre avec une probabilité de 5% ; le choix de la mutation suit le profil d'un mammifère avec 90% de probabilité d'avoir une substitution, 6% une délétion de 4% une insertion.

Les *reads* sont produits à l'aide de générateurs qui simulent la sortie d'une machine particulière. Le générateur choisi ici est SimLord, qui simule la technologie PacBio : chaque nucléotide a 12% de probabilité de subir une insertion, 1% une délétion et 2% une substitution ; cela correspond aux 15% d'erreur de séquençage évoqués précédemment. Les deux groupes de *reads* sont enfin regroupés de telle sorte à pouvoir connaître leur paquet d'origine.

4.2 Application

Pour se placer dans le cas d'application de la méthode évoquée, il faut s'assurer que les *reads* à classifier couvrent tous la même région. L'approximation peut être faite dans le cas de la sélection d'un gène particulier : l'ordre de grandeur d'un gène est de 30 000 nucléotides. Dans le cas contraire, les fréquences d'apparition des *k-mers* parmi les *reads* peuvent être relativement variables. Il est donc probable de devoir moduler les seuils de sélection par rapport à ces fréquences d'apparition.

Pour 240 *reads* issus de deux chaînes de taille 1000, on sélectionne les *k-mers* de taille 10 ayant une fréquence d'apparition comprise entre 25% et 50%. Il s'avère alors que les paires de *k-mers* ayant moins de 40% de *reads* qui ne contiennent aucun des deux *k-mers* partitionnent l'ensemble des *k-mers* en deux groupes de même taille, où chaque paire de *k-mers* de la même catégorie ont une proportion de *reads* supérieure à 40% et chaque paire de *k-mers* de différentes catégories ont une proportion inférieure à 40%. Aussi, les deux groupes formés sont la première moitié de la liste des *k-mers* sélectionnés et la seconde moitié.

La classification ensuite obtenue pour les *reads* forme deux catégories sensiblement de même taille mais qui ne ressemblent en aucun point aux catégories initiales. D'autres essais avec d'autres jeux de données de même taille ont donné les mêmes résultats, tout comme la variation de la taille des *k-mers* (8,10,15,25,40) qui donne les mêmes catégories. En effet, il y a des *k-mers* de taille 40 qui apparaissent au moins dans 25% des *reads* et dans les deux catégories, ce qui est étonnant sachant qu'il y a 15% d'erreur de séquençage, soit une faute tous les 6 nucléotides environ.

Parmi les tentatives de corrections, on peut citer l'utilisation des représentants des *k-mers*. Les *reads* produits par une machine peuvent avoir été retournés. Les *k-mers* extraits sont donc potentiellement soumis à une erreur de sens. Pour éviter cela, on utilise un représentant, qui est le plus petit *k-mer* dans l'ordre lexicographique parmi le *k-mer* lu dans un sens et dans l'autre. Ceci a effectivement réduit le nombre de *k-mer* de 10% environ, sans modifier les groupes finalement obtenus.

L'exploration autour de cette seconde méthode a tout de même montré qu'il est possible avec de tels critères de constituer 3 voire 4 groupes de *k-mers*, qui permettent ensuite de constituer autant de groupes de *reads*. La bonne fusion de ces groupes permet alors de récupérer la classification originale. Néanmoins, tout comme la première méthode, un manque final d'information ne permet pas d'obtenir quelle fusion appliquer parmi celles possibles.

5 Conclusion

On présente deux pistes pour résoudre ce problème de classification des *reads* utilisant des *k-mers*, sous-mots de taille k. Les deux méthodes s'appuient sur une mise en relation de ces *k-mers*. Pour ce faire, on énumère donc les *k-mers* au sein des *reads* sur lesquels on travaille. Une pré-sélection selon la fréquence d'apparition permet d'éviter les *k-mers* parasites, porteurs d'erreurs de mutation ; mais aussi les *k-mers* communs aux deux classes qui ne permettent donc pas de trancher.

Ces *k-mers* sont ensuite traités de manière différente selon les deux méthodes : la première tente de relier ensemble les *k-mers* voisins à l'intérieur d'un graphe. Elle s'appuie sur l'idée qu'il doit être possible de reconstituer un graphe représentant tout le génome. On extrait en-

suite de ce graphe toutes les informations voulues. L'idée a été abandonnée face au manque d'informations nécessaires pour parfaire ce graphe.

La seconde étudie les proportions d'apparition des *k-mers* au sein des *reads* : la présence d'un k-mer ou non au sein d'un read donne de l'information sur ce *read* si l'on classe les k-mers a priori. Cette méthode crée de deux à quatre groupes de *k-mers*. Si la classification en deux groupes ne donne pas de résultats intéressants, la classification en trois ou quatre groupes peut donner la bonne classification en deux catégories sous peine de trouver la bonne fusion.

Les deux méthodes n'ont pas abouti, pour différentes raisons : la première ne présentait pas assez d'information pour conclure, tandis que la seconde ne donnait pas les résultats escomptés. L'utilisation simultanée de ces deux méthodes permettrait peut-être de pouvoir trancher là où il y a eu un manque d'informations. D'autres pistes sont également envisageables. En effet, l'énumération des *k-mers* au sein des *reads* pourrait servir de base de travail à des algorithmes de *pattern mining*.

6 Remerciements

Je tiens tout particulièrement à remercier Lolita Lecompte pour les précisions sur le sujet et Dominique Lavenier pour les discussions que nous avons pu avoir et l'aide apportée à l'écriture de ce rapport.

References

[1] J. Eid et al. "Real-time DNA sequencing from single polymerase molecules". In: *Science* 323 (2009). DOI: 10.1126/science.1162986. URL: https://doi.org/10.1126/science.1162986.

[2] Anthony Rhoads and Kin Fai Au. "PacBio Sequencing and Its Applications". In: *Genomics, Proteomics & Bioinformatics* 13.5 (2015). SI: Metagenomics of Marine Environments, pp. 278–289. ISSN: 1672-0229. DOI: https://doi.org/10.1016/j.gpb.2015.08.002. URL: http://www.sciencedirect.com/science/article/pii/S1672022915001345.

[3] Daniel Branton et al. "The potential and challenges of nanopore sequencing". In: *Nat Biotechnol* 26.10 (Oct. 2008). 18846088[pmid], pp. 1146–1153. ISSN: 1087-0156. DOI: 10.1038/nbt.1495. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2683588/.

[4] Miten Jain et al. "The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community". In: *Genome Biology* 17.1 (Nov. 2016), p. 239. ISSN: 1474-760X. DOI: 10.1186/s13059-016-1103-0. URL: https://doi.org/10.1186/s13059-016-1103-0.

[5] AllSeq. "Pacific Biosciences". In: (). URL: http://allseq.com/knowledgebank/sequencing-platforms/pacific-biosciences.

[6] Jonas Korlach. "Understanding Accuracy in SMRT Sequencing". In: (2013). URL: http://www.pacb.com/wp-content/uploads/2015/09/Perspective_UnderstandingAccuracySMRTSequencing1.pdf.

[7] Erwin L. van Dijk, Yan Jaszczyszyn, and Claude Thermes. "Library preparation methods for next-generation sequencing: Tone down the bias". In: *Experimental Cell Research* 322.1 (2014), pp. 12–20. ISSN: 0014-4827. DOI: https://doi.org/10.1016/j.yexcr.2014.01.008. URL: http://www.sciencedirect.com/science/article/pii/S0014482714000160.

[8] Yesesri Cherukuri and Sarath Chandra Janga. "Benchmarking of de novo assembly algorithms for Nanopore data reveals optimal performance of OLC approaches". In: *BMC Genomics* 17.Suppl 7 (Aug. 2016). 2895[PII], p. 507. ISSN: 1471-2164. DOI: 10.1186/s12864-016-2895-8. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5001211/.